

Duke University Libraries



D01891548-



Digitized by the Internet Archive  
in 2015





DETECTING DIFFERENTIAL SPECIATION RATE IN  
*IPOMOEA* BY A CONTINUOUS-TIME MARKOV MODEL

BY

Dongmei Liu  
Zoology Department  
Duke University

Date: 12/08/2000

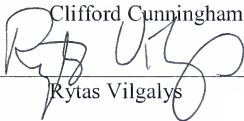
Approved:



Mark D. Rausher, Advisor



Clifford Cunningham



Rytas Vilgalys

A thesis submitted in partial fulfillment of the  
requirements for the degree of Master of Science  
in the department of Zoology  
in the graduate school of  
Duke University

2000



## Abstract

Preliminary study on *Ipomoea* phylogeny shows that species selection may influence flower-color evolution in this genus. Evolutionary transitions between pigmented flower and white flower appear to be highly asymmetrical. White-flowered lineages are less speciose than their pigmented sister clade and evolutionary transitions to white flowers occurred almost exclusively at the tip of phylogeny. A hypothesis to explain this pattern is that species selection acts against white-flowered lineages and prevents them from increasing in relative abundance within the genus *Ipomoea*. A continuous-time Markov model is developed to test differential speciation rates within genus *Ipomoea*. Simulation result shows that the observed distribution of white lineages is not compatible with equal transition rates between pigmented flower and white flower. Only highly asymmetrical transition rates that favor transitions to white flower are compatible with the observed distribution. This result implies that there is a tendency to increase the proportion of white-flowered species in the genus. The only process that could prevent this outcome is species selection or some similar higher-level evolutionary process that acts against white-flowered lineages. The absence of any deep white-flowered clades suggests that it might be white-flowered species have greatly elevated extinction rates compared to pigmented species.





# Table of Contents

Thesis title page	
Abstract title page	
Abstract	iii
Table of contents	iv
List of figures	v
Acknowledgement	vi
I. Introduction	1
II. System	6
III. Method	7
IV. Result	15
V. Discussion	18
Literature cited	23
Figures	27
Appendix	32
1. cuttree.c	32
2. 3-branch.c	39
3. final-tree.c	44
4. sim.c	50
5. probRS.c	57
6. contour.c	67



## List of Figures

Figure 1. Strict consensus tree for <i>Ipomoea</i> based on simultaneous analysis of ITS and <i>waxy</i> sequences.	6
Figure 2. Relationship between R and transition rate. (Simulation is based on linearized trees)	27
Figure 3. Relationship between R and transition rate. (Simulation is based on most parsimonious trees)	28
Figure 4. Relationship between percentage of large R which is bigger than the observed R (Simulation is based on linearized trees)	29
Figure 5. Relationship between percentage of large R which is bigger than the observed R (Simulation is based on most parsimonious trees)	30
Figure 6. Contours which cover 95% of joint probability of R and S (Simulation is based on first 20 most parsimonious trees)	31



## Acknowledgement

I would like to thank Dr. Mark D. Rausher for his help during my Master research. His advice, constructive comments and encouragement helped me throughout the whole project. I would also like to thank Dr. Cliff Cunningham and Dr. Rytas Vilgalys for their support.



## Introduction

It has long been recognized that natural selection may operate at different levels of organization (Lewontin 1970, Wimsatt 1980, Brandon 1982, Sober and Wilson 1994). Evidence has accumulated indicating that besides an individual, the unit of selection may be a gene (Werren et al. 1988), a family group (McCauley 1994), and possibly a population (Wade 1977). More controversial is the notion that selection may operate at higher levels of organization, particularly among lineages or species. Recently, phylogenetic approaches have been used to detect species selection. Statistical analyses of phylogenetic reconstructions can reveal whether clades differ in extinction/speciation rates and whether particular traits are associated with these differences (Mitter et al. 1988, Farrell et al. 1991, Nee et al. 1992, Slowinski and Guyer 1993, Sanderson and Donoghue 1994, 1996). The goal of this project is to examine the role of species selection in influencing the evolution of flower color in morning glories (*Ipomoea*).

The pantropical genus *Ipomoea* L. consists of more than 600 species (Austin and Huaman 1996) of vines, shrubs, and trees that exhibit a tremendous diversity of flower colors. Miller et al. (1999) published a phylogeny of 40 *Ipomoea* species using sequences of the internal transcribed spacer (ITS)





region of nuclear ribosomal DNA and sequences from three exons and two introns of the 3' end of nuclear gene *waxy*. For these analyses, flower colors were grouped into two categories: pigmented and white. Three interesting patterns emerge:

Evolutionary transitions appear to be highly asymmetrical. In particular, parsimony character-state reconstruction of Fig. 1 reveals at least seven, possibly 9, independent transitions from pigmented to white flowers. By contrast, there are at most two, and possibly no, transitions in the reverse direction. (*I. Wrightii*, *I. Graminea*, and *I. imperati* clade may represent reversion.) This pattern suggests that loss of floral pigmentation may be a manifestation of Dollo's law that the loss of complex characters is irreversible (Dollo 1893, Bull and Charnov 1985, Sanderson 1993).

There is a tendency for white-flowered lineages to be less speciose than their pigmented sister clade. In particular, three white-flowered lineages (*I. Saintronanensis* and the mostly white-flowered clade consisting of *I. graminea*, *I. wrightii*, and *I. imperati*) are substantially less speciose than their pigmented sister clades, while no white-flowered lineages are more speciose than their sister clade. The sister taxon of each remaining white-flowered species is a single pigmented species.



Evolutionary transitions to white flowers occurred almost exclusively at the tip of the phylogeny. All white clades are two nodes deep or less, with most represented by a single species.

The strong asymmetry in transitions between pigmented and white flowers can be explained by our current knowledge of the genetic and molecular control of flower color. In *Ipomoea*, as in many plants, floral pigments are anthocyanins, which are produced from the precursors malonyl-CoA and p-coumarl-CoA by a biochemical pathway consisting of six core enzymes (Dooner et al. 1991, van der Meer et al. 1993, Holton and Cornish 1995). Analysis of spontaneous and induced white-flowered mutants in plants, such as *Petunia* and snapdragon (*Antirrhinum*), has revealed that in all cases, white flowers arise whenever one or more structural or regulatory genes of this pathway are deactivated (Forkmann and Dangelmayr 1980, Harker et al. 1990, Inagaki et al. 1994, 1996, Hisatomi et al. 1997, Quattrocchio et al. 1998, Mol et al. 1998). Similarly, naturally occurring white-flowered species and variants, including two in *Ipomoea*, (Ennos and Clegg 1983, Epperson and Clegg 1988) also appear to be due to knockout mutations (Tiffin et al. 1998, Habu et al. 1998, Quattrocchio 1994).



This evidence indicates that white flowers are most likely to arise genetically by loss-of-function mutations affecting specific anthocyanin pathway genes. If this pattern holds upon further investigation, it provides a ready explanation for the asymmetry in flower-color transitions: deactivating mutations are relatively common, whereas restoration of function is rare because, once deactivated, a gene will acquire additional deactivating mutations through genetic drift. These additional mutations will reduce the probability that a single mutation could restore function (but see Hanson et al. 1996).

The apparent strong asymmetry in the direction of flower-color transition, if real, constitutes a process that would tend to increase the proportion of white-flowered species in the genus. If unopposed by other processes, it would eventually produce a genus in which most species have white flowers. Because the asymmetry results from the integration of all evolutionary processes acting below the species level, the only process that could prevent this outcome is species selection or some similar higher-level evolutionary process. Moreover, the observation that transitions to white flowers occur almost exclusively at the tips of our phylogeny suggests that white-flowered species, for unknown ecological reasons, have greatly elevated extinction rates, compared to pigmented species, though the concomitant existence of decreased speciation rates in white-flowered lineages can not be ruled out. The



overall goal of the project is to test the hypothesis that species selection acts against white-flowered lineages and prevents them from increasing in relative abundance.





# Systems

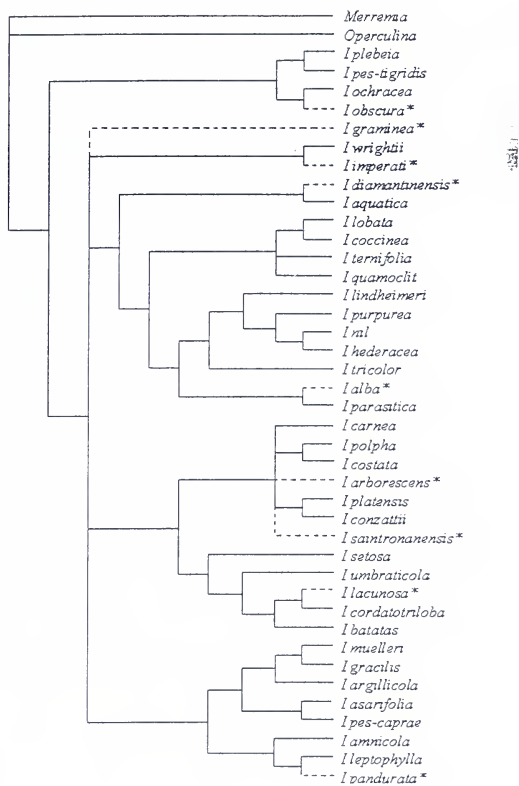


Fig. 1 Strict consensus tree for *Ipomoea* based on simultaneous analysis of ITS and waxy sequences (Miller, 1999). \* indicate white flower species.



## Methods

In order to test the asymmetry of *Ipomoea* phylogeny, we developed the following method. A set of equally most parsimonious phylogenies is generated. One phylogeny is randomly selected for analysis and linearized. Branch lengths are then estimated by maximum likelihood under the assumption for a molecular clock. Based on this linearized phylogeny, the distribution of white-flowered species is reconstructed using a continuous-time Markov model. Using this model, 1000 trees with random flower-color transitions will be generated stochastically for each phylogeny. We choose R (number of white species/total length of white lineages) and S (number of white species) to be the two statistics to test asymmetrical distribution of white-flowered and pigmented flowered species in phylogeny. This procedure is repeated for a number of randomly-chosen equally parsimonious topologies with a set of transition rates between white flowered species and pigmented flowered species. It yields a distribution of joint probabilities of R and S. The statistics are also calculated for the actual tree and compared to the frequency distribution of the statistics among the simulations. If fewer than five percent of the statistics from the stochastic simulations are as extreme as that for the actual tree, the stochastic model may be rejected as incompatible with the observed pattern.



(1) Phylogeny reconstruction.

The phylogeny of 40 species was analyzed using the sequence of the internal transcribed spacer (ITS) region of nuclear ribosomal DNA and sequences for three exons and two introns of the 3' end of the nuclear gene *waxy*. *Merremia tuberosa* and *Operculina brownii* are used as outgroups. Combined ITS and *waxy* sequence has 1285 nucleotides. The phylogeny was reconstructed by PAUP using parsimony. Among the 1285 sites, 249 sites are parsimony-informative. Finally, we obtained 56 equally most-parsimonious phylogenies.

(2) Generate linearized phylogeny and estimate branch length by maximum likelihood method.

Randomly select one maximum parsimonious phylogeny for analysis. Takezaki et al. (1995) proposed a method to sequentially prune species exhibiting significant deviation from the average nucleotide substitution rate until rate variation among the remaining species is consistent with a molecular clock. Their method is called two-cluster test. The principle of the test is to examine the equality of the average substitution rate for two clusters that are created by a node in a given tree. The basic quantities they care about are the averages of observed numbers of substitutions per site from node to the tips of



the two clusters. Under the assumption of rate constancy, the expected difference of the two clusters' substitution rates is zero. If the test shows significant rate difference between the two clusters, they eliminate the cluster whose average branch length from the root is more different from the average of all sequences than the other cluster. This test and sequence elimination therefore proceeds from terminal nodes to the root of the tree. A linearized tree will then be constructed for a given topology under the assumption of rate constancy. Takezaki has written a program package to generate linearized tree. I used this package to linearize phylogenies in this project.

Branch lengths are then estimated by maximum likelihood method under the assumption of a molecular clock. Yang (1997) wrote a package of programs, called PAML, for phylogenetic analyses of DNA sequences using the maximum likelihood method. General assumptions of the programs are that substitutions occur independently in different lineages and among sites, the process of substitution is a time-homogeneous Markov process, and the frequencies of nucleotides have remained constant over the time period covered by data. I used this program package to obtain branch lengths of trees in this project.

(3) Simulate transitions between pigmented flower and white flower.





Pagel (1994) developed a continuous-time Markov model for character evolution. The model is designed to estimate simultaneous transition rates in pairs of binary characters arrayed on a phylogeny. All species in the study could have only one of two characters, in our case, a pigmented flower or a white flower. I assign it to be either P state (pigmented flower) or W state (white flower). Along a branch beginning with state P/W, the flower color will either remain the same or change to the other state. The probability for this change or unchange over time  $t$  ( $t$  is the branch length) is  $P_{pp}(t)$ ,  $P_{ww}(t)$ ,  $P_{wp}(t)$ , or  $P_{pw}(t)$ . Assume that the flower color evolves according to a Markov process such that the probability of change is independent in each branch of the phylogenetic tree, and that the probability of changing from one state to the next depends only upon the state at the beginning of the branch, not on events previous to that. The probability that flower color changes from state  $i$  to state  $j$  over time interval  $t+dt$  is given by

$$P_{ij}(t+dt)=P_{ii}(t)*q_{ij}*dt+P_{ij}(t)*(1-q_{ij})*dt$$

where  $t$  is an arbitrary time period,  $dt$  is a short interval of time, and  $q_{ij}$  is transition rate parameter denoting the rate of change from state  $i$  to state  $j$  over an infinitesimally short time interval. The  $1-q_{ij}$  term represents the probability of staying in state  $j$ . This equation states that flower color can change from state  $i$  to state  $j$  either by remaining unchanged for a period  $t$  followed by a



transition to state  $j$ , or by changing to state  $j$  during time period  $t$ , then remaining in state  $j$  over period  $dt$ . The set of equations describing the four possible probabilities of interest can be presented in matrix form

$$P(t+dt)=P(t)*(I+Qdt),$$

$$\begin{bmatrix} 1-P_{pw}(t+dt) & P_{pw}(t+dt) \\ P_{wp}(t+dt) & 1-P_{wp}(t+dt) \end{bmatrix} = \begin{bmatrix} P_{pp}(t) & P_{pw}(t) \\ P_{wp}(t) & P_{ww}(t) \end{bmatrix} * \begin{bmatrix} (1-q_{pw})dt & q_{pw}dt \\ q_{wp}dt & (1-q_{wp})dt \end{bmatrix}$$

To solve for the  $P(t)$  in terms of the rate parameters in  $Q$ ,

$$dP(t)/dt=P(t)*Q,$$

$$P(t)=\exp(Qt),$$

$P(t)$  is only a function of the transition rates between pigmented flower and white flower.

Each simulation begins with a pigmented flowered species. Two transition rates are assigned at the beginning of the simulation. Along each branch, there is an expected transition probability,  $P(t)$ , calculated based on the continuous-time Markov model. The program could generate random numbers uniformly distributed in  $(0,1)$ . By comparing the random number to the expected  $P(t)$ , the stochastic model decides whether flower color would change along the branch. Beginning with a pigmented flower species on the root of a tree, it would finally generate a white flower species distribution on the tips of the tree based



on the given phylogeny. By assigning different transition rates, I could obtain distributions of white-flowered species corresponding to different transition rates.

#### (4) Choice of test statistics

The primary goal of this project is to develop a statistical test for whether white lineages are confined to the tips of robustly supported clades, which would imply higher rates of extinction (or lower rates of speciation) in white-flowered lineages. We want to find a statistic which could capture the degree to which white-flowered lineages are confined to the tips of the phylogeny. This test statistic should reflect two properties that are negatively correlated: if the probability of transition from pigmented to white is low,  $S$ , the number of white species will be low, but the ratio  $R = (\text{number of white species}) / (\text{total length of white lineages})$  will be high; by contrast, if that transition probability is high, the number of white species will be high, but  $R$  will be low. Because a single test statistic is not likely to capture this relationship, we believe that a bivariate plot of  $R$  vs.  $S$  will provide a useful test of whether the observed distribution of the white lineages differs from what expected under the stochastic model.



In the simulation, the program could remember on which branch the transition between white flower and pigmented flower happened and the character state (white or pigmented flower) of each node. If a transition from pigmented flower to white flower happens on a branch, I assume the transition happened on the base of the branch. When calculating the total length of white lineages, I trace back each white flowered species to its oldest ancestor whose character state is white flowered. The total branch length between that ancestor and the white flowered species is the length of this white lineage. The total length of white lineages is the summation of length of all white lineages subtracting the branches which are counted repeatedly.

(5) Find the probability contour which covers 95% of the simulated values.

For each phylogeny, repeat the stochastic simulations described above for 1000 times. It will yield a joint probability distribution of R and S. On the boarder of the region that covers all simulated values, the point which has the minimum joint probability will be cut off. This process is continued until the remaining region encloses 95% of the joint probabilities. It will finally generate a set of probability contours corresponding to different levels of transition rate.





Steps (2)-(5) are then repeated for 56 equally most parsimonious trees generated in step A. This will yield an average distribution of joint probability of R and S. We believe this mean can be taken as a reasonable estimation of overall probability. The statistic, R and S, is also calculated for the reference tree (Fig. 1). Since the reference tree is probably the one most close to the actual tree, the reference tree's R and S are taken to be observed R and S. If the observed values of these two variables fall outside the probability contour that encloses 95% of the simulated values, the observed distribution of white lineages can be taken as incompatible with the corresponding transition rates. There are two cases we want to test: equal transition rate between pigmented and white flowers or a rate of zero for gain of pigmentation.



## Result

Among the 42 species we studied, 9 species are white-flowered. They are *I. obscura*, *I. graminea*, *I. imperati*, *I. diamantinensis*, *I. alba*, *I. arborescens*, *I. saintronaensis*, *I. lacunosa*, and *I. pandurata*. The branch length of Fig. 1 is also estimated by PAML. The observed R ratio (number of white species)/(totally length of white lineages) is 63.87. The actual S (number of white species) is 9.

After linearization, some of the most parsimonious trees became the same. I finally obtained a total of 18 linearized trees. The largest tree has 28 species, while the smallest one has 17 species. The 18 linearized trees' branch lengths were then estimated by PAML. Simulations were done along all linearized trees and the original most parsimonious trees as well. The transition rate from pigmented flower to white flower is set to be from 1 to 45 ( $=100 \times q_{pw}$ , refer to Methods section), while the transition rate from white flower to pigmented flower is set to be from 0 to 45 ( $=100 \times q_{wp}$ ). Because all simulations begin with a pigmented flower species on the root of a tree, the transition rate from pigmented flower to white flower can not be zero.



The result is presented in 3 sections. Fig. 2 and Fig. 3 are three dimensional graphs showing the relationships between  $R$  and the two transition rates. X axis is the transition rate from white flower to pigmented flower, Y axis is the transition rate from pigmented flower to white flower, and Z axis is average  $R$  value of 1000 simulations. Fig. 2 is the average simulation result of all linearized trees. Fig. 3 is the average simulation result of all most parsimonious trees. Both Fig. 2 and Fig. 3 are separated by the number of white species. From Fig. 2, we see that in all 4 cases, simulation tends to get big  $R$  when transition rate from white flower to pigmented flower is low and transition rate from pigmented flower to white flower is high.  $R$  value decreases along with increasing transition rate from white flower to pigmented flower and decreasing transition rate in reverse direction. Among the 4 cases, when the number of white species is small,  $R$  tends to distribute in a large range, from about 75 to 20 in Fig. 2.1, and when the number of white species is big,  $R$  distributes in a small range, from 60 to 40. Both Fig. 2 and Fig. 3 show the same pattern. In both Fig. 2.1 and Fig. 3.1,  $R$  approaches the observed value of 63.87 only when rate of transition from pigmented flower to white flower is high and reverse transition rate is low.

Fig. 4 and Fig. 5 are also three dimensional graphs showing the relationships between  $R$  and the two transition rates. X and Y axes are still the same with



Fig. 2 and Fig. 3, while Z axis is average percentage of R value which is bigger than the observed R in 1000 simulations. Fig. 4 is the average simulation result of all linearized trees. Fig. 5 is the average simulation result of all most parsimonious trees. Again, both Fig. 4 and Fig. 5 are separated by number of white species. From Fig. 4, we see that in all 4 cases, when transition rate from white-flowered species to pigmented-flowered species is low and transition rate in reverse direction is high, we could expect to see more than 5 percent of simulations get R bigger than observed R. When the two transition rates are equal or transition rate from white-flowered species to pigmented-flowered species is bigger than the transition rate in reverse direction, almost no simulations get an R value bigger than the observed R. Fig. 5 shows the same pattern. Again, Fig. 4 and Fig. 5 show range of R tends to shrink along with the increasing of number of white species.

Fig. 6 shows the contours that cover 95% of joint probabilities of R and S. X axis is S, number of white species. Y axis is R, (number of white species)/(total length of white lineages). The little triangle indicates the observed R and S. Only in Fig. 6.2 and Fig. 6.3, the contour covers the observed values. In these two cases, the transition rate from white flower to pigmented flower is much less than the transition rate in reverse direction. Another pattern we could see from Fig. 6 is that along the diagonal from upper





right corner to bottom left corner, transition probability from pigmented flower to white flower decreases from very big to very small, along this change,  $R$  tends to decrease and  $S$  tends to increase.



## Discussion

From the three figures, Fig. 4, Fig. 5, and Fig. 6, we can see the null hypothesis of equal transition rate is rejected. The observed pattern is only compatible with a high transition rate from pigmented flowered to white flower and low transition rate from white flower to pigmented flower. This confirms our original hypothesis that deactivating mutations are relatively common, whereas restoration of function is rare. Since this strong asymmetry in the direction of flower-color transition constitutes a process that would tend to increase the proportion of white flower species in the genus, there must be a process, such as species selection or some similar higher-level evolutionary process that could prevent this outcome. This indicates a higher rate of extinction or lower rate of speciation in white-flowered lineages. The absence of deep white-flower clades suggests that white-flowered species might have greatly elevated extinction rates compared to pigmented species.

From Fig. 2 to Fig. 5, we see that the range of  $R$  tends to decrease when  $S$  increases. This is because when the number of white species is small, white species could be on either short terminal branches or long terminal branches, so the variance of total length of white lineages tends to be large; when the number of white species increases, for example, 25 out of 42 species are



white-flowered species, the white lineages will trace back to the root of the tree and total length of white lineages does not change very much. So  $R$  is constricted in a small range.

Fig. 6 shows an interesting result. Fig. 6.3 indicates that when transition rate from pigmented flower to white flower is much higher than transition rate from white flower to pigmented flower, the average number of white species is small, around 13. While Fig. 6.7 shows that when transition rate from pigmented flower to white flower is much less than transition rate in reverse direction, the average number of white species is big, around 25. This contradicts the expectation that higher transition probability from pigmented flower to white flower will generate a bigger number of white-flowered species.

Intuitively, we expect to see higher transition rate from pigmented flower to white flower will result in increased mean number of white flowered species. But in Fig. 6, as the rate of transition from white flower to pigmented flower increases (e.g. going down a column), the mean number of white species increases and as the rate of transition from pigmented to white increases (e.g. going across a row), the mean number of white-flowered species decreases. When the transition rate from white flower to pigmented flower is fixed,



increasing the transition rate from pigmented flower to white flower (going across a row) will increase the probability of transition from pigmented flower to white flower on each single branch. However, the small number of white flowered species is more likely to be due to multiple transitions on the external branch, while the big number of white flowered species is more likely to be due to a single transition on the internal branch which is close to the root of the phylogeny. When the probability of transition on each single branch is increased, it will benefit the first case more than the second one, since more branches are involved in the first case. Multiple origin of white flowered species on external branches will gain more weight. This will drag down the mean number of white flowered species.

When the transition rate from pigmented flower to white flower is fixed, increasing the transition rate from white flower to pigmented flower (going down a column) will increase the probability of losing a white flowered species on each single branch. In order to clean up a pigmented flower to white flower transition which happened on an internal branch close to the root of the phylogeny, back transition on the branch right next to the origin of white flowered species is important. If this step is missed, it will become harder and harder to clean up white flowered species, since they will spread out in the following branching process. In this case, cleaning up white





flowered species is more likely to be a single hit. For multiple origin of white flowered species on external branches, the probability to clean up white flowered species will increase in fold of number of transitions to white. It will be affected more than the former case, since it is more likely to be multiple hits. When transition rate from white flower to pigmented flower increases, big number of white flowered species will gain more weight. This will increase mean of number of white flowered species.

After Miller et al. (1999) published the phylogeny of 40 *Ipomoea* species, this analysis has been expanded by adding 30 additional taxa. Repeating the same statistical test to the new phylogeny would give a more robust conclusion. Also, ancestral state reconstruction based on preliminary phylogeny suggests that there is a trend toward white-flowered clades being less speciose than pigmented sister clades. However, because most white lineages are confined to tip branches, the sister clade of most species contain only one pigmented species each. A denser sampling of species closely related to these white species may reveal sister clades containing more than one species. The inclusion of the additional taxa should allow us to ascertain whether the hypothesis of white-flowered clades being less speciose than pigmented sister clades and should allow a more reliable comparison of sister-clade diversity.



## Literature cited

- Austin, D. F., and Z. Huaman. 1996. A synopsis of *Ipomoea* (Convolvaceae) in the Americas. *Taxon* 45: 3-38
- Brandon, R. N. 1982. The levels of selection. pp. 315-323 in P. Asquith and T. Nickles (eds). *PSA 1982, Vol. 1. Philosophy of Science Association*, East Lansing, MI.
- Bull, J. J., and E. L. Charnov. 1985. On irreversible evolution. *Evolution* 39:1149-1155.
- Dollo, L. 1983. Les Lois de l'évolution. *Bull. Belge. Geol.* 7:164-167.
- Dooner, H. K., T. P. Robbins and R. A. Jorgensen. 1991. Genetic and developmental control of anthocyanin biosynthesis. *Annual Review of Genetics* 25:173-199.
- Ennos, R. A., and M. T. Clegg. 1983. Flower color variation in the morning glory, *Ipomoea purpurea*. *Journal of Heredity* 74:247-250.
- Epperson, B. K., and M. T. Clegg. 1988. Genetics of flower color polymorphism in the common morning glory (*Ipomoea purpurea*) *Journal of Heredity* 79:64-68.
- Farrell, B. D. Dussourd, and C. Mitter. 1991. Escalation of plant defenses: Do latex and resin canals spur plant diversification? *The American Naturalist* 138: 881-900.
- Forkmann, G., and B. dangelmayr. 1980. Genetic control of chalcone isomerase activity in flowers of *Dianthus caryophyllus*. *Biochemical Genetics* 18: 519-527.
- Habu, Y., Y. Hisatomi, and S. Iida. 1998. Molecular characterization of the mutable *flaked* allele for flower variegation in the common morning glory. *The Plant Journal* 16: 371-376.
- Hanson, M. A., B. S. Gaut., A. O. Stec, S. I. Fuerstenberg, M. M. Goodman, E. H. Coe, J. F. Doebley. 1996. Evolution of anthocyanin biosynthesis in maize kernels: The role of regulatory and enzymatic loci. *Genetics* 143: 1395-1407.



Harker, C. L., T. H. Noell Ellis and Enrico S. Coen. 1990. Identification and genetic regulation of the chalcone synthase multigene family in pea. *The Plant Cell* 2: 185-194.

Hisatomi, Y., K. Hanada, and S. Iida. 1997. The retrotransposon Rtp1 is integrated into a novel type of minisatellite, MiniSip1, in the genome of the common morning glory and carries another new type of minisatellite, MiniSip2. *Theoretical and Applied Genetics* 95: 1049-1056.

Holton, T. A., and E. C. Cornish. 1995. Genetics and biochemistry of anthocyanin biosynthesis. *The Plant Cell*. 7: 1071-1083.

Inagaki, Y., Y. Hisatomi, T. Suzuki, K. Kasahara, and S. Iida. 1994. Isolation of a suppressor mutator/enhancer-like transposable element, Tpn1, from Japanese morning glory bearing variegated flowers. *The Plant Cell* 6: 375-383.

Inagaki, Y., Y. Hisatomi, and S. Iida. 1996. Somatic mutations caused by excision of the transposable element, *Tpn1*, from the DFR gene for pigmentation in sub-epidermal layer of periclinally chimeric flowers of Japanese morning glory and their germinal transmission to their progeny. *Theoretical and Applied Genetics* 92: 499-504.

Lewontin, R. C. 1970. The units of selection. *Annals Review of Ecology and Systematics* 1; 1-18.

McCualey, D. E. 1994. Intrademic group selection imposed by a parasitoid-host interaction. *The American naturalist* 144; 1-13.

Miller, R. E., M. D. Rausher, and P. S. Manos. 1999. Phylogenetic systematics of *Ipomoea* (Convolvulaceae) based on ITS and *waxy* sequences. *Systematic Botany* 24: 209-227.

Mitter, C., B. Farrell, and B. Wiegmann. 1988. The phylogenetic study of adaptive zones: Has phytophagy promoted insect diversification? *The American Naturalist* 132: 107-128.

Mol, J., E. Grotewold, and R. Koes. 1998. How genes paint flowers and seeds. *Trends in Plant Science* 3: 212-217.

Nee, S., A. C. Mooers, and P. H. Harvey. 1992. Tempo and mode of evolution revealed from molecular phylogenies. *Proceedings of the National Academy of Sciences* 89: 8322-8326.



Pagel, M. 1994. Detecting correlated evolution on phylogenies: a general method for the comparative analysis of discrete characters. *Proceedings of the Royal Society of London B* 255: 37-45.

Quattrocchio, F. 1994. Regulatory genes controlling flower pigmentation in *Petunia hybrida*. Ph.D. Dissertation. Vrije Universiteit, Amsterdam, Netherlands.

Quattrocchio, F., and J. F. Wing, K. v. d. Woude, J. N. m. mol, and R. Koes. 1998. Analysis of bHLH and MYB domain proteins: species-specific regulatory differences are caused by divergent evolution of target anthocyanin genes. *The Plant Journal* 13: 475-488.

Sanderson, M. J. 1993. Reversibility in evolution; a maximum likelihood approach to character gain/loss bias in phylogenies. *Evolution* 47: 236-252.

Sanderson, M. J., and M. J. Donoghue. 1994. Shifts in diversification rates with the origin of angiosperms. *Science* 264: 1590-1593.

Sanderson, M. J., and M. J. Donoghue. 1996. Reconstructing shifts in diversification rate on phylogenetic trees. *Trends in Ecology and Evolution* 11: 15-20.

Slowonski, J. B., and C. Guyer. 1993. Testing whether certain traits have caused amplified diversification: An improved method based on a model of random speciation and extinction. *The American Naturalist* 142: 1019-1024.

Sober, E., and D. S. Wilson. 1994. A critical review of philosophical work on the units of selection problem. *Philosophy of science* 61: 534-555.

Takezaki, N., A. Rzhetsky, and M. Nei 1995. Phylogenetic test of the molecular clock and linearized trees. *Molecular Biology and Evolution* 12: 823-833.

Tiffin, P., R. E. Miller, and M. D. Rausher. 1998. Control of expression patterns of anthocyanin structural genes by two loci in the common morning glory. *Genes and Genetic Systems* 73: 105-110.

van der Meer, I. M., A. R. Stuitje and J. N. M. Mol. 1993. Regulation of general phenylpropanoid and flavonoid gene expression. in D. P. S. verma, ed. *Control of Plant Gene Expression*, pp. 125-155. CRC Press.





Wade, M. J. 1977. An experimental study of group selection. *Evolution* 31: 134-153.

Werren, J. H., U. Nur, and C.-I. Wu. 1988. Selfish genetic elements. *Trends in Ecology and Evolution* 3: 297-302.

Wimsatt, W. C. 1980. Reductionistic research strategies and their biases in the units of selection controversy. pp. 213-259 in T. Nickles (ed.). *Scientific Discovery, Volume II, historical and Scientific Cases Studies*. Reidel, Dordrecht.

Yang, Z. 1997. PAML: a program package for phylogenetic analysis by maximum likelihood. *CABIOS* 13: 555-556.



Fig. 2 R vs. transition rate  
(Linearized trees)

Fig. 2.1 number of white species =5

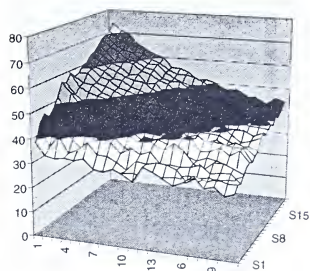


Fig. 2.2 number of white species =10

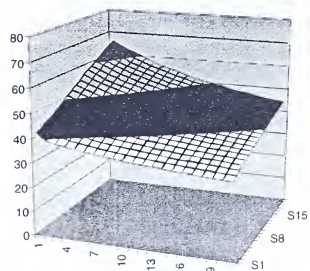


Fig. 2.3 number of white species =15

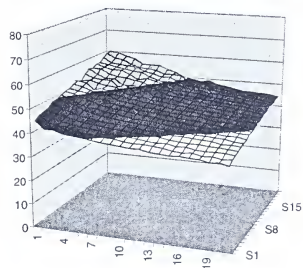


Fig. 2.4 number of white species =20

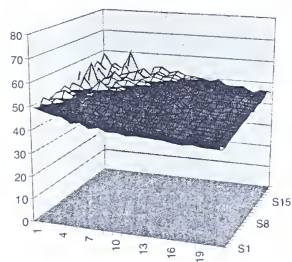


Fig. 2 X axis is the transition rate from white flower to pigmented flower, Y axis is the transition rate from pigmented flower to white flower, Z axis is the simulated R.



Fig.3 R vs. transition rate  
(Most parsimonious trees)

Fig. 3.1 number of white species =10

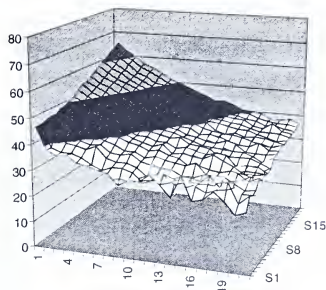


Fig. 3.2 number of white species =15

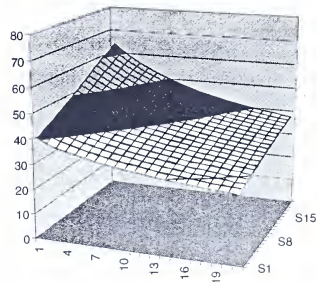


Fig. 3.3 number of white species =20

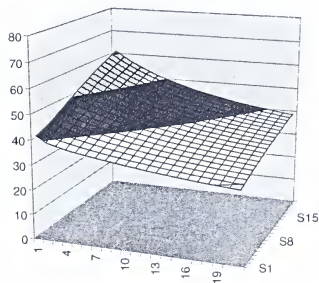


Fig. 3.4 number of white species =25

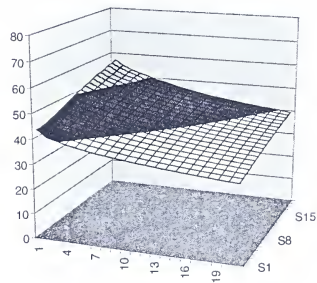


Fig. 3 X axis is the transition rate from white flower to pigmented flower, Y axis is the transition rate from pigmented flower to white flower, Z axis is the simulated R.



THE JOURNAL OF THE AMERICAN MEDICAL ASSOCIATION  
PUBLISHED WEEKLY  
CHICAGO, ILL., MAY 1, 1919  
Vol. 27, No. 18

Fig. 6 Contours which cover 95% of joint probability of  $X$  and  $Y$  (simulation based on first 20 most parsimonious trees)

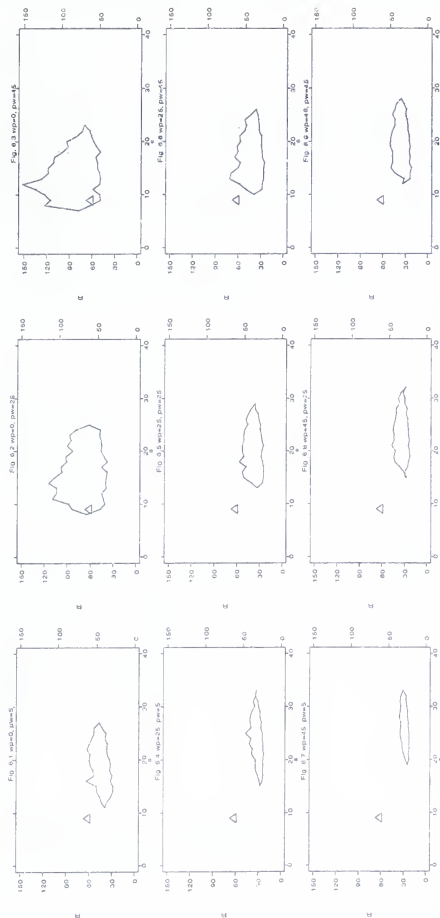


Fig. 6 wp is the transition rate from white flower to pigmented flower, pw is the transition rate from pigmented flower to white flower. In each figure, X axis is S, the number of white species, Y axis is R, ratio of number of white species to total length of white lineages. The little triangle indicate the observed (R,S).





## Appendix

### 1. cuttree.c

```
/* Nei's program will tell which species/nodes to cut off.
This program generates the new tree after cutting off. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int i,j,k,l,m,ncut,cut[100],N,tp[150],nodeA[100],nodeB[100];
int ntp,mother,grandma;
int ncutspecies,cutspecies[100],nsite,check,newN,newtp[150];
int ancestor[150];
int nleft,nright;
char left,right;
char seq[50][2000],species[50][50];

FILE *inp,*inputseq,*out,*outseq,*inputname;

main()
{
    input();
    calculation();
    output();
}

input()
{
    /* input parsimonious tree */

    inp=fopen("cuttertree.dat", "r");
    fscanf(inp,"%d\n",&N); /* N, number of species */

    k=0;
    l=0;
    ntp=0;
    do{
        ntp++;
        fscanf(inp,"%d\t",&tp[ntp]);
        if(tp[ntp]==1000) k=k+1;
        if(tp[ntp]==2000) l=l+1;
    }while(tp[ntp]!=5000);
    ntp--;
    if(k!=l) printf("datafile is wrong.\n");
    fscanf(inp,"%\n");
}
```



```

/* input the nodes/species to cut off */

fscanf(inp,"%d\n",&ncut);
/* ncut, number of species/nodes to cut off */

for(i=1;i<=ncut;i++)
{
    fscanf(inp,"%d\n",&cut[i]);
}

fclose(inp);

/* input sequences data */

inputseq=fopen("sequence.dat", "r");
fscanf(inputseq,"%d\n",&N);          /* N, number of
species */
fscanf(inputseq,"%d\n",&nsite);      /* nsite, total
number of sites
                                compared among sequences */
for(i=1;i<=N;i++)
{
    fscanf(inputseq,"%d\n",&m);
    for(j=1;j<=nsite;j++)
    {
        seq[m][j]=getc(inputseq);
        /* seq[m][j], the mth sequence's jth site
*/
    }
}

fclose(inputseq);

inputname=fopen("species_name","r");
fscanf(inputname,"%d\n",&N);

for(i=1;i<=N;i++)
{
    fscanf(inputname,"%d ",&j);
    fscanf(inputname,"%s\n",species[j]);
}

fclose(inputname);

}

calculation()
{
    tree_structure();
    cut_tree();

```



```

}

tree_structure()
{

    i=0;
    k=N;
    do{

        /* find the right side of the node */

        do{
            i=i+1;
        }while(tp[i]!=2000); /* 2000 indicate right edge of a
cluster */
        k=k+1; /* k is the node # */
        tp[i]=k; /* reassign right edge to be node#
*/
        nodeB[k]=tp[i-1];

        /* find the left side of the node */

        j=i;
        do{
            j=j-1;
        }while(tp[j]!=1000); /* 1000 indicates left edge of a
cluster */
        tp[j]=k; /* reassign left edge to be node#
*/
        nodeA[k]=tp[j+1];

    }while(i<ntp);

    for(i=1;i<=3*N-2;i++)
    {
        newtp[i]=tp[i];
    }

    for(i=N+1;i<=2*N-2;i++)
    {
        k=nodeA[i];
        ancestor[k]=i;
        k=nodeB[i];
        ancestor[k]=i;
    }

}

cut_tree()
{

```



```

        }
        else
        {
            nodeB[grandma]=nodeB[mother];
        }
    }
    else
    {
        nodeB[mother]=0;
        ancestor[nodeA[mother]]=grandma;
        if (mother==nodeA[grandma])
        {
            nodeA[grandma]=nodeA[mother];
        }
    }
    else
    {
        nodeB[grandma]=nodeA[mother];
    }
}
}
}

newN=N-ncutspecies;

}

```

```

output()
{

    out=fopen("baseml_tree.dat","w");
    fprintf(out,"%d\t%d\n\n",newN,1);

    left='(';
    right=')';
    nleft=0;
    nright=0;

    for(i=1;i<=ntp;i++)
    {
        if((tp[i]>N)&&(newtp[i]!=0))
        {
            check=0;
            for(j=1;j<i;j++)
            {
                if(tp[j]==tp[i]) check=1;
            }
            if (check==0)
            {
                fprintf(out,"%c",left);
                nleft++;
            }
        }
    }
}

```





```

        }
        else
        {
            fprintf(out,"%c",right);
            nright++;
        }
    }
    if ((tp[i]<=N)&&(newtp[i]!=0))
    {
        if ((tp[i-1]<=N)&&(newtp[i-1]!=0))
        fprintf(out,"%c",',');
        fprintf(out,"%s",species[tp[i]]);
    }
}
if(nleft!=nright) printf("There is sth wrong.\n");

fclose(out);

outseq=fopen("baseml_seq.dat","w");
fprintf(outseq,"%d\t",newN);
fprintf(outseq,"%d\n",nsite);

for(i=1;i<=N;i++)
{
    check=0;
    for(j=1;j<=ncutspecies;j++)
    {
        if(i==cutspecies[j]) check=1;
    }
    if(check==0)
    {
        fprintf(outseq,"%s\n",species[i]);
        for(j=1;j<=nsite;j++)
        {
            fprintf(outseq,"%c",seq[i][j]);
        }
        fprintf(outseq,"\n");
    }
}
fclose(outseq);
}

```



## 2. 3-branch.c

```

/* Change trees with 3 branches at a single node into
bifurcating trees */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int N, tp[150], i, j, k, l, nodeA[100], nodeB[100], nodeC[100];
int ntp, check, bad_node;
int left, right, nleft, nright, newtp[150], newntp;
char species[50][50];
FILE *inputtp, *outtp;

main()
{
    input();
    structure();
    output();
}

input()
{
    /* input parsimonious tree */

    inputtp=fopen("topology.dat", "r");
    fscanf(inputtp, "%d\n", &N);          /* N. number of
species */

    k=0;
    l=0;
    i=0;
    do{
        i=i+1;
        fscanf(inputtp, "%d\t", &tp[i]);
        if(tp[i]==1000) k=k+1;
        if(tp[i]==2000) l=l+1;
    }while(tp[i]!=5000);
    ntp=i-1;
    if(k!=l) printf("datafile is wrong.\n");

    fclose(inputtp);
}

structure()
{

```



```

i=0;
k=N;
do{

    /* find the right side of the node */

    do{
        i=i+1;
    }while(tp[i]!=2000); /* 2000 indicate right edge of a
cluster */
    k=k+1;                /* k is the node # */
    tp[i]=k;              /* reassign right edge to be node#
*/
    nodeB[k]=tp[i-1];

    /* find the left side of the node */

    j=i;
    do{
        j=j-1;
    }while(tp[j]!=1000); /* 1000 indicates left edge of a
cluster */
    tp[j]=k;              /* reassign left edge to be node#
*/
    nodeA[k]=tp[j+1];

    /*check if this tree have 3 node case */

    l=j+1;
    if((tp[l]<=N)&&(tp[l+1]!=nodeB[k]))
    {
        nodeC[k]=tp[l+1];
    }

    if(tp[l]>N)
    {
        do{
            l=l+1;
        }while(tp[l]!=nodeA[k]); /* find the right edge of
nodeA */

        if(tp[l+1]!=nodeB[k]) /* 3 node case */
        {
            nodeC[k]=tp[l+1];
        }
    }

    if (nodeC[k]!=0)
    {
        if (nodeC[k]<=N) l=l+1;
        else

```



```

        {
            l=l+1;
            do{
                l=l+1;
            }while(tp[l]!=nodeC[k]);
        }

        if(tp[l+1]!=nodeB[k])
            printf("There is sth wrong for node %d\n",k);
    }

}while(i<ntp);

i=N;
check=0;
do{
    i=i+1;
    if(nodeC[i]!=0)
    {
        check=1;
        bad_node=i;
    }
}while(i<=2*N-1);

    for(i=1;i<=ntp;i++)
    {
        newtp[i]=tp[i];
    }
newntp=ntp;

    if(check==1) new_tree();
}

new_tree()
{
    for(i=2*N-2;i>bad_node;i--)
    {
        nodeA[i]=nodeA[i-1];
        if(nodeA[i]>=bad_node) nodeA[i]++;
        nodeB[i]=nodeB[i-1];
        if(nodeB[i]>=bad_node) nodeB[i]++;
    }
    nodeA[bad_node+1]=bad_node;
    nodeB[bad_node+1]=nodeB[bad_node];
    nodeB[bad_node]=nodeC[bad_node];

    for(i=1;i<=ntp;i++)
    {
        if(newtp[i]>=bad_node) newtp[i]++;
    }
}

```





```

i=0;
check=0;
do{
    i=i+1;
    if (tp[i]==bad_node) check=1;
}while (check==0);
left=i;

for (j=ntp+1;j>left;j--)
{
    newtp[j]=newtp[j-1];
}
newtp[left+1]=bad_node;
k=left;
check=0;
do{
    k++;
    if (newtp[k]==nodeC[bad_node]) check++;
}while (check<2);

for (j=ntp+2;j>k;j--)
{
    newtp[j]=newtp[j-1];
}
newtp[k+1]=bad_node;

newntp=ntp+2;
}

output()
{
    outtp=fopen("n_topology.dat", "w");
    fprintf(outtp, "%d\n", N);

    left=1000;
    right=2000;
    nleft=0;
    nright=0;

    for (i=1;i<=newntp;i++)
    {
        if (newtp[i]>N)
        {
            check=0;
            for (j=1;j<i;j++)
            {
                if (newtp[j]==newtp[i]) check=1;
            }
            if (check==0)
            {
                fprintf(outtp, "%d\t", left);
            }
        }
    }
}

```



```

        nleft++;
    }
    else
    {
        fprintf(outtp,"%d\t",right);
        nright++;
    }
}
if(newtp[i]<=N)
{
    fprintf(outtp,"%d\t",newtp[i]);
}
}
fprintf(outtp,"%d\n",5000);
if(nleft!=nright)
{
    printf("There is sth wrong.\n");
    printf("nleft=%d\n",nleft);
    printf("nright=%d\n",nright);
}

fclose(outtp);
}

```



### 3. final-tree.c

```
/* read in branch length from baseml, generate the input file
for sim.c */
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
```

```
int N,i,j,k,l,m,newtvp[150],newntvp,nbr,orderbr[150],check;
int nodeA[100],nodeB[100];
int len,nodeC[150],bad_node;
float inpbr[150],br[150];
char tp[1000],c;
char species[50][50];
```

```
FILE *inp,*out,*inputtp;
```

```
main()
```

```
{
```

```
    input();
    calculation();
    output();
```

```
}
```

```
input()
```

```
{
```

```
    inp=fopen("mlb.dat","r");
    fscanf(inp,"%d\n\n",&N);
```

```
    i=0;
    m=0;
    do{
        i++;
        tp[i]=getc(inp);
```

```
        if(tp[i]==':')
        {
            m++;
            fscanf(inp,"%f",&inpbr[m]);
        }
    }while(tp[i]!=';');
```

```
    nbr=m;
    
```

```
/* check out input */
```

```
/*
```



```

printf("%d\n\n",N);
i=0;
m=0;
do{
    i++;
    printf("%c",tp[i]);
    if(tp[i]!='')
    {
        m++;
        printf("%8.6f",inpbr[m]);
    }

}while(tp[i]!='');
printf("\n");*/

fclose(inp);
}

```

```

calculation()
{
    change_input();
    structure();
    branch();
}

```

```

change_input()
{
    j=0; /* newtp */
    i=0; /* tp */
    m=0; /* species */

    do{
        i++;
        if(tp[i]=='(')
        {
            j++;
            newtp[j]=1000;
        }

        if(tp[i]==')')
        {
            j++;
            newtp[j]=2000;
        }

        if((tp[i]>='A')&&(tp[i]<='Z'))
        {

```





```

        m++;
        k=i;
        do{
            k++;
        }while(tp[k]!=':');

        for (l=1;l<=k-i;l++)
        {
            len=strlen(species[m]);
            species[m][len]=tp[l+i-1];
        }

        j++;
        newtp[j]=m;
    }

    }while(tp[i]!=':');
    newntp=j;
}

structure()
{
    i=0;
    k=N;
    do{

        /* find the right side of the node */

        do{
            i=i+1;
        }while(newtp[i]!=2000); /* 2000 indicate right edge of
a cluster */
        k=k+1; /* k is the node # */
        newtp[i]=k; /* reassign right edge to be
node# */
        nodeB[k]=newtp[i-1];

        /* find the left side of the node */

        j=i;
        do{
            j=j-1;
        }while(newtp[j]!=1000); /* 1000 indicates left edge of
a cluster */
        newtp[j]=k; /* reassign left edge to be
node# */
        nodeA[k]=newtp[j+1];

        /*check if this tree have 3 node case */

```



```

l=j+1;
if((newtp[l]<=N)&&(newtp[l+1]!=nodeB[k]))
{
    nodeC[k]=newtp[l+1];
}

if(newtp[l]>N)
{
    do{
        l=l+1;
    }while(newtp[l]!=nodeA[k]); /* find the right edge
of nodeA */

    if(newtp[l+1]!=nodeB[k]) /* 3 node case */
    {
        nodeC[k]=newtp[l+1];
    }
}

if(nodeC[k]!=0)
{
    if(nodeC[k]<=N) l=l+1;
    else
    {
        l=l+1;
        do{
            l=l+1;
        }while(newtp[l]!=nodeC[k]);
    }

    if(newtp[l+1]!=nodeB[k])
    printf("There is sth wrong for node %d\n",k);
}

}while(i<newnttp);

/*for(i=1;i<=newnttp;i++)
{
    printf("%d\t",newtp[i]);
}*/

}

branch()
{
    m=0;
    for(i=1;i<=newnttp;i++)
    {
        if(newtp[i]<=N)
        {

```



```

        m++;
        orderbr[m]=i;
    }
    if (newtp[i]>N)
    {
        check=0;
        for (j=i-1;j>=1;j--)
        {
            if (newtp[j]==newtp[i]) check=1;
        }
        if (check==1)
        {
            m++;
            orderbr[m]=i;
        }
    }
}

for (i=1;i<=nbr;i++)
{
    br[newtp[orderbr[i]]]=inpbr[i];
}

/*
for (i=1;i<=newntp;i++)
{
    if (newtp[i]<=N)
    {
        printf("%s:%8.6f",species[newtp[i]],br[newtp[i]]);
    }
    if (newtp[i]>N)
    {
        check=0;
        for (j=i-1;j>=1;j--)
        {
            if (newtp[j]==newtp[i]) check=1;
        }
        if (check==1)
        {
            printf("%c:%8.6f",' ' ,br[newtp[i]]);
        }
        if (check==0)
        {
            printf("%c",'(');
        }
    }
}*/
}

output()
{

```



```

    out=fopen("sim.dat","w");
    fprintf(out,"%d sequences\n",N);

    for(i=1;i<=N;i++)
    {
        fprintf(out,"%d",i);
        fprintf(out," %s\n",species[i]);
    }

    check=0;
    for(i=N+1;i<=2*N-1;i++)
    {
        if(nodeC[i]!=0)
        {
            check=1;
            bad_node=i;
        }
    }

    fprintf(out,"bad_node\t%d\n",bad_node);

    for(i=N+1;i<=2*N-3;i++)
    {
        fprintf(out,"%3d and %3d",i,nodeA[i]);
        fprintf(out,"    %lf\n",br[nodeA[i]]);

        fprintf(out,"%3d and %3d",i,nodeB[i]);
        fprintf(out,"    %lf\n",br[nodeB[i]]);

        if(i==bad_node)
        {
            fprintf(out,"%3d and %3d",i,nodeC[i]);
            fprintf(out,"    %lf\n",br[nodeC[i]]);
        }
    }

    if(bad_node==0)
    {
        i=2*N-2;
        fprintf(out,"%3d and %3d",i,nodeA[i]);
        fprintf(out,"    %lf\n",br[nodeA[i]]);
        fprintf(out,"%3d and %3d",i,nodeB[i]);
        fprintf(out,"    %lf\n",br[nodeB[i]]);
        fprintf(out,"%3d and    1\t %lf\n",i,br[1]);
    }
    else
    {
        i=2*N-3;
        fprintf(out,"%3d and    1\t %lf\n",i,br[1]);
    }
    fclose(out);
}

```





```

4. sim.c

/* simulate transition between purple flower and white flower,
find the final distribution of
purple(-1) flower species and white(1) flower species*/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define step 40

int i,j,k,l,m,w,newN,nodeA[100],nodeB[100],nodeC[100];
int ancestor[100],color[100];
int
check,seed,nw[5000],newseed,runtime,nmutation,mutation[100];
int bad_node,wp,pw;
int order[5000],y[100],printposition,totaly,count[100];
int biggerthanR[50][50];
float totalR[50][50];
double
branch[150],rate,p_wrate,w_prate,t,p,sample,ttw[5000],R[5000];
double minimum_x,minimum_y,maximum_x,maximum_y;
double step_x,orderR[5000],minimum_R;
double x[100],percenty[100],ordery[100];
char species[50][50],string[10];

FILE *inp,*out;

main()
{
    input();
    for(wp=0;wp<=45;wp++)
    {
        w_prate=(float)(wp)/100;
        for(pw=1;pw<=45;pw++)
        {
            p_wrate=(float)(pw)/100;
            totalR[wp][pw]=0;
            printf("w_prate=%f\tp_wrate=%f\n",w_prate,p_wrate);
            for(runtime=1;runtime<=1000;runtime++)
            {
                calculation();
            }
            record();
        }
    }
    output();
}

input()

```



```

{
    printf("Seed for random number generator (samller than 6
digits): \n");
    scanf("%6d", &seed);
    srand48(seed);

    inp=fopen("sim.dat","r");
    fscanf(inp,"%d sequences\n",&newN);

    for(i=1;i<=newN;i++)
    {
        fscanf(inp,"%d",&j);
        fscanf(inp," %s\n",species[j]);
    }
    fscanf(inp,"%s\t%d\n",string,&bad_node);

    for(i=newN+1;i<=2*newN-3;i++)
    {
        fscanf(inp,"%3d and %3d",&j,&nodeA[i]);
        fscanf(inp," %lf\n",&branch[nodeA[j]]);

        fscanf(inp,"%3d and %3d",&j,&nodeB[i]);
        fscanf(inp," %lf\n",&branch[nodeB[j]]);

        if(i==bad_node)
        {
            fscanf(inp,"%3d and %3d",&j,&nodeC[i]);
            fscanf(inp," %lf\n",&branch[nodeC[j]]);
        }
    }

    if(bad_node==0)
    {
        i=2*newN-2;
        fscanf(inp,"%3d and %3d",&j,&nodeA[i]);
        fscanf(inp," %lf\n",&branch[nodeA[j]]);
        fscanf(inp,"%3d and %3d",&j,&nodeB[j]);
        fscanf(inp," %lf\n",&branch[nodeB[j]]);
        fscanf(inp,"%3d and %3d %lf\n",&j,&branch[l]);
    }
    else
    {
        i=2*newN-3;
        fscanf(inp,"%3d and %3d %lf\n",&j,&branch[l]);
    }

    fclose(inp);
}

calculation()

```



{

```
if (bad_node==0)
{
    for (i=newN+1;i<=2*newN-2;i++)
    {
        k=nodeA[i];
        ancestor[k]=i;
        k=nodeB[i];
        ancestor[k]=i;
    }

    m=2*newN-2;
    color[m]=-1;
    for (i=1;i<=2*newN-3;i++)
    {
        color[i]=0;
    }

    nmutation=0;
    do{
        k=nodeA[m];
        t=branch[k];
    } while (color[m]==-1)
    {
        p=w_prater-p_wrate*exp(-1*(w_prater+p_wrate)*t);
        p=p/(w_prater+p_wrate);
        sample=drand48();
        if (sample>p)
        {
            color[k]=1;
            nmutation=nmutation+1;
            mutation[nmutation]=k;
        }
    }
    else
    {
        color[k]=-1;
    }
}
if (color[m]==1)
{
    p=p_wrate+w_prater*exp(-1*(w_prater+p_wrate)*t);
    p=p/(w_prater+p_wrate);
    sample=drand48();
    if (sample>p)
    {
        color[k]=-1;
        nmutation=nmutation+1;
        mutation[nmutation]=k;
    }
}
else
{
```



```

        color[k]=1;
    }
}

m=k;
}while (m>newN) ;

do{
    k=ancestor[m];
    if (k!=bad_node)
    {
        if (color[nodeA[k]]==0) down();
        else
        {
            if (color[nodeB[k]]==0) down();
        }
    }
    else
    {
        if (color[nodeA[k]]==0) down();
        else
        {
            if (color[nodeB[k]]==0) down();
            else
            {
                if (color[nodeC[k]]==0) down();
            }
        }
    }
    m=k;
}while (m<2*newN-4);

}

/* check */
check=0;
l=1;
if (bad_node==0)
{
    do{
        l=l+1;
        if (color[l]==0) check=1;
    }while (l<2*newN-2 && check==0);
}
else
{
    do{
        l=l+1;
        if (color[l]==0) check=1;
    }while (l<2*newN-3 && check==0);
}

if (check==1)

```





```

        {
            printf("calculation is not completed.\n");
            printf("species %d is missing.\n",l);
        }

    statistics();

}

down()
{
    do{
        if(color[nodeA[k]]==0) l=nodeA[k];
        else
        {
            if(color[nodeB[k]]==0) l=nodeB[k];
            else
            {
                if((k==bad_node)&&(color[nodeC[k]]==0))
l=nodeC[k];
                else printf("sth is wrong at node %d\n",k);
            }
        }

        t=branch[l];
        if(color[k]==-1)
        {
            p=w_prate-p_wrate*exp(-1*(w_prate+p_wrate)*t);
            p=p/(w_prate+p_wrate);
            sample=drand48();
            if(sample>p)
            {
                color[l]=1;
                nmutation=nmutation+1;
                mutation[nmutation]=l;
            }
        }
        else
        {
            color[l]=-1;
        }
    }
    if(color[k]==1)
    {
        p=p_wrate+w_prate*exp(-1*(w_prate+p_wrate)*t);
        p=p/(w_prate+p_wrate);
        sample=drand48();
        if(sample>p)
        {
            color[l]=-1;
            nmutation=nmutation+1;
            mutation[nmutation]=l;
        }
    }
}

```



```

        else
        {
            color[l]=1;
        }
    }
    k=1;
}while (k>newN);
}

statistics()
{
    for(i=1;i<=2*newN-1;i++)
    {
        count[i]=0;
    }

    ttw[runtime]=0;
    nw[runtime]=0;
    for(i=1;i<=newN;i++)
    {
        if(color[i]==1)
        {
            nw[runtime]=nw[runtime]+1;
            w=i;
            do{
                ttw[runtime]=ttw[runtime]+branch[w];
                count[w]=1;
                w=ancestor[w];
            }while((color[w]==1)&&(count[w]==0));
        }
    }
    if(ttw[runtime]==0) R[runtime]=0;
    else R[runtime]=nw[runtime]/ttw[runtime];
    totalR[wp][pw]=totalR[wp][pw]+R[runtime];
}

record()
{
    for(i=1;i<=12;i++)
    {
        y[i]=0;
    }

    for(i=1;i<=1000;i++)
    {
        if(R[i]<30) y[1]++;
        if(R[i]>80) y[12]++;
        if((R[i]<=80)&&(R[i]>=30))
        {

```



```

        j=(int)((R[i]-30)/5)+2;
        y[j]++;
    }
}

output()
{
    out=fopen("final.out","w");
    for(wp=1;wp<=20;wp++)
    {
        for(pw=1;pw<=20;pw++)
        {
            fprintf(out,"%f\t",totalR[wp][pw]/1000);
        }
        fprintf(out,"\n");
    }
    fclose(out);
}

```



## 5. probRS.c

```

/* get joint probability of R and S, based on all parsimony
trees */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define step 20

int i,j,k,l,m,w,newN,nodeA[100],nodeB[100],nodeC[100];
int ancestor[100],color[100];
int
check,seed,nw[6000],newseed,runtime,nmutation,mutation[100];
int bad_node,wp,pw;
int count[150],nwhite[50][45][45];
int nR[110][50][40][40];
int len,nf,nfile,ct[110][50][45][45];
float prob_R_given_S[110][50][40][40],prob_S[50][40][40];
float joint_prob_R_S[110][50][45][45];
float probRS[110][50][45][45],del;
float boarder,boarder2,sum[40][40];
double
branch[150],rate,p_wrate,w_prate,t,p,sample,ttw[6000],R[6000];
char species[50][50],string[10];
char
filename[30]={ 'R','_','S','_','w','p','_','p','w','_','\0' };
char add[15]={ '0','1','2','3','4','5','6','7','8','9','\0' };
char newfilename[30];
char file[80][50];

FILE *inp,*out;

main()
{
    getfile();
    set_zero();
    for (nf=1;nf<=nfile;nf++)
    {
        printf("computing tree %d\n",nf);
        input();
        for (wp=0;wp<=40;wp++)
        {
            w_prate=(float) (wp)/100;
            for (pw=1;pw<=40;pw++)
            {
                p_wrate=(float) (pw)/100;

                printf("w_prate=%f\tp_wrate=%f\n",w_prate,p_wrate);
                setup();
                for (runtime=1;runtime<=1000;runtime++)

```





```

        {
            calculation();
        }
        prob();
    }
}

output();

}

getfile()
{
    printf("Seed for random number generator (samller than 6
digits): \n");
    scanf("%6d", &seed);
    srand48(seed);

    inp=fopen("tree_input_files", "r");
    fscanf(inp, "%d\n", &nfile);

    for(i=1; i<=nfile; i++)
    {
        fscanf(inp, "%s\n", file[i]);
    }
    fclose(inp);
}

set_zero()
{
    for(wp=0; wp<=40; wp++)
    {
        for(pw=1; pw<=40; pw++)
        {
            for(i=1; i<=newN; i++)
            {
                for(j=0; j<=100; j++)
                {
                    ct[j][i][wp][pw]=0;
                    probRS[j][i][wp][pw]=0;
                }
            }
        }
    }
}

```



```

input()
{
    inp=fopen(file[nf],"r");
    fscanf(inp,"%d sequences\n",&newN);

    for(i=1;i<=newN;i++)
    {
        fscanf(inp,"%d",&j);
        fscanf(inp,"%s\n",species[j]);
    }
    fscanf(inp,"%s\t%d\n",string,&bad_node);

    for(i=newN+1;i<=2*newN-3;i++)
    {
        fscanf(inp,"%3d and %3d",&j,&nodeA[i]);
        fscanf(inp,"%1f\n",&branch[nodeA[j]]);

        fscanf(inp,"%3d and %3d",&j,&nodeB[i]);
        fscanf(inp,"%1f\n",&branch[nodeB[j]]);

        if(i==bad_node)
        {
            fscanf(inp,"%3d and %3d",&j,&nodeC[i]);
            fscanf(inp,"%1f\n",&branch[nodeC[j]]);
        }
    }

    if(bad_node==0)
    {
        i=2*newN-2;
        fscanf(inp,"%3d and %3d",&j,&nodeA[i]);
        fscanf(inp,"%1f\n",&branch[nodeA[j]]);
        fscanf(inp,"%3d and %3d",&j,&nodeB[j]);
        fscanf(inp,"%1f\n",&branch[nodeB[j]]);
        fscanf(inp,"%3d and %1\t %1f\n",&j,&branch[l]);
    }
    else
    {
        i=2*newN-3;
        fscanf(inp,"%3d and %1\t %1f\n",&j,&branch[l]);
    }

    fclose(inp);
}

setup()
{
    for(i=1;i<=newN;i++)

```



```

    {
        nwhite[i][wp][pw]=0;
        for(j=0;j<=100;j++)
        {
            nR[j][i][wp][pw]=0;
        }
    }

}

calculation()
{

    if(bad_node==0)
    {
        for(i=newN+1;i<=2*newN-2;i++)
        {
            k=nodeA[i];
            ancestor[k]=i;
            k=nodeB[i];
            ancestor[k]=i;
        }

        m=2*newN-2;
        color[m]=-1;
        for(i=1;i<=2*newN-3;i++)
        {
            color[i]=0;
        }

        nmutation=0;
        do{
            k=nodeA[m];
            t=branch[k];
        }if(color[m]==-1)
        {
            p=w_prate-p_wrate*exp(-1*(w_prate+p_wrate)*t);
            p=p/(w_prate+p_wrate);
            sample=drand48();
            if(sample>p)
            {
                color[k]=1;
                nmutation=nmutation+1;
                mutation[nmutation]=k;
            }
        }else
        {
            color[k]=-1;
        }
    }
    if(color[m]==1)
    {

```



```

p=p_wrate+w_prate*exp(-1*(w_prate+p_wrate)*t);
p=p/(w_prate+p_wrate);
sample=drand48();
if(sample>p)
{
    color[k]=-1;
    nmutation=nmutation+1;
    mutation[nmutation]=k;
}
else
{
    color[k]=1;
}
}

m=k;
}while(m>newN);

do{
    k=ancestor[m];
    if(k!=bad_node)
    {
        if(color[nodeA[k]]==0) down();
        else
        {
            if(color[nodeB[k]]==0) down();
        }
    }
    else
    {
        if(color[nodeA[k]]==0) down();
        else
        {
            if(color[nodeB[k]]==0) down();
            else
            {
                if(color[nodeC[k]]==0) down();
            }
        }
    }
    m=k;
}while(m<2*newN-4);

}

/* check */
check=0;
l=1;
if(bad_node==0)
{
    do{
        l=l+1;
        if(color[l]==0) check=1;
    }
}

```





```

        }while(l<2*newN-2 && check==0);
    }
else
{
    do{
        l=l+1;
        if(color[l]==0) check=1;
    }while(l<2*newN-3 && check==0);
}

if(check==1)
{
    printf("calculation is not completed.\n");
    printf("species %d is missing.\n",l);
}

statistics();

}

down()
{
    do{
        if(color[nodeA[k]]==0) l=nodeA[k];
        else
        {
            if(color[nodeB[k]]==0) l=nodeB[k];
            else
            {
                if((k==bad_node)&&(color[nodeC[k]]==0))
l=nodeC[k];
                else printf("sth is wrong at node %d\n",k);
            }
        }

        t=branch[l];
        if(color[k]==-1)
        {
            p=w_prate-p_wrate*exp(-1*(w_prate+p_wrate)*t);
            p=p/(w_prate+p_wrate);
            sample=drand48();
            if(sample>p)
            {
                color[l]=1;
                nmutation=nmutation+1;
                mutation[nmutation]=l;
            }
        }
        else
        {
            color[l]=-1;
        }
    }
}

```



```

        if(color[k]==1)
        {
            p=p_wrate+w_prate*exp(-1*(w_prate+p_wrate)*t);
            p=p/(w_prate+p_wrate);
            sample=drand48();
            if(sample>p)
            {
                color[l]=-1;
                nmutation=nmutation+1;
                mutation[nmutation]=l;
            }
            else
            {
                color[l]=1;
            }
        }
        k=k+1;
    }while(k>newN);
}

statistics()
{
    for(i=1;i<=2*newN-1;i++)
    {
        count[i]=0;
    }

    ttw[runtime]=0;
    nw[runtime]=0;
    for(i=1;i<=newN;i++)
    {
        if(color[i]==1)
        {
            nw[runtime]=nw[runtime]+1;
            w=i;
            do{
                ttw[runtime]=ttw[runtime]+branch[w];
                count[w]=1;
                w=ancestor[w];
            }while((color[w]==1)&&(count[w]==0));
        }
    }

    if(ttw[runtime]!=0) R[runtime]=nw[runtime]/ttw[runtime];
    else R[runtime]=0;

    nwhite[nw[runtime]][wp][pw]++;
    if((int)R[runtime]<100)
    nR[(int)R[runtime]][nw[runtime]][wp][pw]++;
}

```



```

}

prob()
{
    for(i=1;i<=newN;i++)
    {
        prob_S[i][wp][pw]=(float)nwhite[i][wp][pw]/runtime;
        for(j=0;j<=100;j++)
        {
            if(nwhite[i][wp][pw]!=0)
            {
                prob_R_given_S[j][i][wp][pw]=(float)nR[j][i][wp][pw]/nwhite[i][wp][pw];

                joint_prob_R_S[j][i][wp][pw]=prob_R_given_S[j][i][wp][pw]*prob_S[i][wp][pw];
                ct[j][i][wp][pw]++;
                del=joint_prob_R_S[j][i][wp][pw]-
                probRS[j][i][wp][pw];
                probRS[j][i][wp][pw]=probRS[j][i][wp][pw]+del/ct[j][i][wp][pw];
            }
        }

        /*test*/
        /*
        boarder=0;
        for(i=1;i<=newN;i++)
        {
            for(j=0;j<=100;j++)
            {
                boarder=boarder+joint_prob_R_S[j][i][wp][pw];
            }
        }
        printf("boarder=%f\n",boarder);*/
    }
}

output()
{
    for(wp=0;wp<=40;wp++)
    {
        for(pw=1;pw<=40;pw++)
        {
            sum[wp][pw]=0;
            for(i=1;i<=newN;i++)
            {

```



```

        for (j=0;j<=100;j++)
        {
            sum[wp] [pw]=sum[wp] [pw]+probRS[j] [i] [wp] [pw];
        }
    }
    /*printf("total prob of wp(%d) pw(%d)
    =%f\n",wp,pw,sum[wp] [pw]);*/
    }

    for (wp=0;wp<=40;wp++)
    {
        for (pw=1;pw<=40;pw++)
        {
            for (i=1;i<=newN;i++)
            {
                for (j=0;j<=100;j++)
                {

                    probRS[j] [i] [wp] [pw]=probRS[j] [i] [wp] [pw]/sum[wp] [pw];
                }
            }
        }
    }

    for (i=0;i<=8;i++)
    {
        wp=i*5;
        for (j=1;j<=8;j++)
        {
            pw=j*5;
            strcpy(newfilename, filename);
            len=strlen(newfilename);
            k=wp/10;
            newfilename[len]=add[k];
            l=wp%10;
            newfilename[len+1]=add[l];
            newfilename[len+2]='_';
            k=pw/10;
            newfilename[len+3]=add[k];
            l=pw%10;
            newfilename[len+4]=add[l];
            newfilename[len+5]='\0';
            printf("%s\n",newfilename);
            write();
        }
    }

}

write()
{

```





```

out=fopen(newfilename,"w");
for(k=1;k<=newN;k++)
{
    for(l=0;l<=100;l++)
    {
        fprintf(out,"%f\t",probRS[l][k][wp][pw]);
    }
    fprintf(out,"\n");
}
fclose(out);
}

```



```

6. contour.c

/* find the area which covers 95% of all simulate (R,S) */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int
i,j,k,bottom[50],up[50],cut,R,S,smallestS,biggestS,bad_S,check;
int wp,pw,len,l,w_p,p_w;
float probRS[210][50],newprobRS[210][50];
float min,total_prob;
char
filename[30]={'R','_','S','_','w','_','p','_','_','p','_','w','_','_','\0'};
char add[15]={'0','1','2','3','4','5','6','7','8','9','\0'};
char inputfile[30];
char writefile[30]={'c','o','n','\0'};
char outputfile[30];

FILE *inp,*out;

main()
{
    for(w_p=0;w_p<=9;w_p++)
    {
        wp=w_p*5;
        for(p_w=1;p_w<=9;p_w++)
        {
            pw=p_w*5;
            strcpy(inputfile,filename);
            len=strlen(inputfile);
            k=wp/10;
            inputfile[len]=add[k];
            l=wp%10;
            inputfile[len+1]=add[l];
            inputfile[len+2]='_';
            k=pw/10;
            inputfile[len+3]=add[k];
            l=pw%10;
            inputfile[len+4]=add[l];
            inputfile[len+5]='\0';
            printf("%s\n",inputfile);

            input();
            setup();
            total_prob=1;
            do{
                boarder();
                cut_off();
                find_bad_S();
            }while(1);
        }
    }
}

```



```

        /*printf("total_prob=%f\n",total_prob);*/
    }while(total_prob>0.95);

    strcpy(outputfile,writefile);
    len=strlen(outputfile);
    k=wp/10;
    outputfile[len]=add[k];
    l=wp%10;
    outputfile[len+1]=add[l];
    k=pw/10;
    outputfile[len+2]=add[k];
    l=pw%10;
    outputfile[len+3]=add[l];
    outputfile[len+4]='\0';
    printf("%s\n",outputfile);

    output();
}

}

input()
{

    inp=fopen(inputfile,"r");
    for(i=1;i<=42;i++)
    {
        for(j=0;j<=200;j++)
        {
            fscanf(inp,"%f\t",&probRS[j][i]);
        }
        fscanf(inp,"\n");
    }
    fclose(inp);

}

setup()
{

    for(i=1;i<=42;i++)
    {
        for(j=0;j<=200;j++)
        {
            newprobRS[j][i]=probRS[j][i];
        }
    }

}

```



```

boarder()
{
    for(i=1;i<=42;i++)
    {
        j=0;
        do{
            j++;
        }while((newprobRS[j][i]==0)&&(j<=200));
        if(j<=200)
        {
            bottom[i]=j;
            j=201;
            do{
                j--;
            }while(newprobRS[j][i]==0);
            up[i]=j;
        }
        else
        {
            bottom[i]=0;
            up[i]=0;
        }
    }

    /*printf("bottom[%d]=%d\tup[%d]=%d\n",i,bottom[i],i,up[i]);*/
}

cut_off()
{
    min=2;
    for(i=1;i<=42;i++)
    {
        if(bottom[i]+up[i]!=0)
        {
            if(min>newprobRS[bottom[i]][i])
            {
                min=newprobRS[bottom[i]][i];
                cut=bottom[i]*100+i;
            }
            if(min>newprobRS[up[i]][i])
            {
                min=newprobRS[up[i]][i];
                cut=up[i]*100+i;
            }
        }
    }

    R=cut/100;
}

```





```

S=cut%100;
newprobRS[R][S]=0;
total_prob=total_prob-min;
/*printf("cut off prob[%d] [%d]
%f\ttotal_prob=%f\n",R,S,probRS[R][S],total_prob);*/
}

find_bad_S()
{
    do{
        j=0;
        do{
            j++;
        }while(bottom[j]+up[j]==0);

        smallestS=j;
        j=43;
        do{
            j--;
        }while(bottom[j]+up[j]==0);
        biggestS=j;

        check=0;
        j=smallestS;
        do{
            j++;
            if(bottom[j]+up[j]==0)
            {
                check=1;
                bad_S=j;
            }
        }while((check==0)&&(j<biggestS));

        if(check==1)
        {
            if(bad_S-smallestS<biggestS-bad_S)
            {
                for(i=smallestS;i<=bad_S;i++)
                {
                    for(k=bottom[i];k<=up[i];k++)
                    {
                        newprobRS[k][i]=0;
                        total_prob=total_prob-newprobRS[k][i];
                    }
                    bottom[i]=0;
                    up[i]=0;
                }
            }
            else
            {

```



```

        for (i=biggestS;i>=bad_S;i--)
        {
            for (k=bottom[i];k<=up[i];k++)
            {
                newprobRS[k][i]=0;
                total_prob=total_prob-newprobRS[k][i];
            }
            bottom[i]=0;
            up[i]=0;
        }
    }

/*printf("smallestS=%d\tbiggestS=%d\tbad_S=%d\n",smallestS,bigg
estS,bad_S);*/
    }while(check==1);

}

output()
{
    out=fopen(outputfile,"w");
    fprintf(out,"%d\t\t%d\n",9,63);
    for(i=1;i<=42;i++)
    {
        if(bottom[i]!=0)
            fprintf(out,"%d\t%d\n",i,bottom[i]);
    }
    for(i=42;i>=1;i--)
    {
        if(bottom[i]!=0)
            fprintf(out,"%d\t%d\n",i,up[i]);
    }
    fclose(out);
}

```

453 R8 61725  
03/21/01 33357









